CS 330 – Spring 2023 – Homework 08

Due: Friday 4/21 at 11:59 pm on Gradescope

Collaboration policy Collaboration on homework problems is permitted, you are allowed to discuss each problem with at most 3 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For dynamic programming algorithms, follow the solution guideline laid out in class:

1. precisely define the sub-problem with proper indexing
2. give the recursive formula and argue about its correctness
3. write the DP algorithm
4. write an algorithm that prints the elements in the optimal solution

You may use anything we learned in class without further explanation. This includes using algorithms from class as subroutines, stating facts that we proved in class, e.g. correctness of subroutines, running time of subroutines and use the notation. Your description should be at the level that it is clear to a classmate who is taking the course with you.

You should be as clear and concise as possible in your write-up of solutions.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

(*) It is fine if the English description concentrates on the high level ideas and doesn't include all the details. But the reader should not have to figure out your solution solely based on the pseudocode. You can also add comments to your pseudocode, in fact that is best practice.

Problem 1. (Bead chain) You are given a long chain of colored beads. At first the chain looks random to you, but after looking at it for a little bit you see some patterns emerge. Specifically, you notice that there are parts in the chain that are symmetric; the order of colors from left to right is the same as right to left. Excited by your discovery you decide to cut up your beads into shorter symmetric chains. Your goal is to do so using as few cuts as possible so that each segment is symmetric. (Note that a single bead is considered to be a symmetric chain.) You cannot change the order of beads.

1. (Not to be handed in) Here is an example of chain of beads, try to find the optimal cuts.

   [red, blue, green, blue, red, red, red, red, red, red, red, red, blue, yellow, blue, red]

2. As input you are given the sequence of n beads in an array B. To make your life easier you are also given an $n \times n$ table S of True/False values. $S[i][j]$ is True if the chain starting from bead i to j is symmetric, false otherwise. Note that only half of S is filled as $i \leq j$.

   Design a DP algorithm that finds a division of the chain into symmetric chains using the minimum number of cuts. Your algorithm should return both the number of cuts as well as the subchains themselves (i.e. the location of the cuts).

3. In the previous part we said that S is given as part of the input. Now we ask you to find an algorithm that given B computes the values of S. (*This problem can be solved by an iterative algorithm, you don't have to use DP for it.*)

Problem 2. (Stock broker) You are a stock broker trading stocks on your clients' behalf. The client tells you the type of stock that they have and what they want to get instead. It's your job to make the trade in a cost-efficient way - possible over a sequence of multiple trades. You may assume that you are always trading a single stock of one type for a single stock of another type. Some stocks are more expensive than others. If you trade stock a for stock b, then you pay a positive amount if b is more expensive than a, and pay negative amount, i.e. earn some money, if b is cheaper than a.

1. As input you are given the $n \times n$ table C of trading costs. $C[a][b]$ is the amount you pay or earn if you directly exchange a single stock of a for b. (Note that $C[a][b]$ is positive if you have to pay, negative if you earn money on the transaction.) Every index of C is known.

   Design a DP algorithm that returns the $n \times n$ table D which contains the most optimal trading costs between pairs of stocks. That is, $D[x][y]$ is the least cost of exchanging the stock x for y over a sequence of trades. The values may be positive or negative or 0. For full credit your algorithm should run in time $O(n^3)$. (*Hint: consider the min cost trading sequence making only use of stocks 1...k*)

   For this assignment define the subproblems, find and prove the recursive formula (no induction) and the DP algorithm complete with runtime analysis. You can omit the backtracking algorithm.

2. In real life there is no such thing as infinite money making! But, observe that you can run your DP algorithm regardless whether the input contains a negative weight trading cycle. Suppose that somebody has computed the minimum costs in the DP table D for this problem. Using D explain how to decide in $O(n)$ time whether the input contains a negative weight cycle. (It's sufficient to give a one-two sentence verbal description of your approach and justification why it returns the correct value.)